INTECH
open science | open minds

OPEN ACCESS
ARTICLE

International Journal of Advanced Robotic Systems

Heterogeneous Context-Aware Robots Providing a Personalized Building Tour

Regular Paper

Anna Hristoskova[1,*], Carlos E. Agüero[2], Manuela Veloso[3] and Filip De Turck[1]

1 Department of Information Technology, Ghent University - IBBT, Ghent, Belgium
2 Robotics Group, Universidad Rey Juan Carlos, Madrid, Spain
3 Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
* Corresponding author E-mail: anna.hristoskova@intec.ugent.be

Received 30 May 2012; Accepted 31 Oct 2012

DOI: 10.5772/54797

Abstract Existing robot guides offer a tour of a building, such as a museum or science centre, to one or more visitors. Usually the tours are predefined and lack support for dynamic interactions between the different robots.

This paper focuses on the distributed collaboration of multiple heterogeneous robots (receptionist, companion) guiding visitors through a building. Semantic techniques support the formal definition of tour topics, the available content on a specific topic, and the robot and person profiles including interests and acquired knowledge. The robot guides select topics depending on their participants' interests and prior knowledge. Whenever one guide moves into the proximity of another, the guides automatically exchange participants, optimizing the amount of interesting topics.

Robot collaboration is realized through the development of a software module that allows a robot to transparently include behaviours performed by other robots into its own set of behaviours. The multi-robot visitor guide application is integrated into an extended distributed heterogeneous robot team, using a receptionist robot that was not originally designed to cooperate with the guides. Evaluation of the implemented algorithms presents a 90% content coverage of relevant topics for the participants.

Keywords Semantic Web, Multi-Robot Interaction, Context-Awareness, Task Transparency, Robot Behaviour

1. Introduction

Applications supporting multiple robots require the simultaneous achievement of complex interdependent tasks. Such systems focus on techniques related to distributed robot coordination and task allocation. The individual robots should be able to execute several tasks independently and optimize the task execution through seamless collaboration with other robots. These autonomous robot interactions require exchange of context between the robots. Context is defined as the dialogue between the robot and its participants, for example on visited places and personal details.

The emerging trend is the use of ontologies to define context. An ontology is a formal specification of an agreed conceptualization of a domain in the context of knowledge description. The semantic robot profiles

specify a vocabulary that can be used during the robot interactions. One example is the Robot Ontology for Urban Search and Rescue profile [1], which captures information on robots and their capabilities in a search and rescue emergency scenario. The ontology in [2] focuses on exploiting models in the activities of a multi-robot system, such as the ability of a robot to pass through a door, rotate on the spot and park. One challenge is the optimal selection of the relevant information to minimize the amount of data exchanged between the robots.

The scenario of a robot tour guide requires one robot leading a group of participants. Mimicking a real person, the robot should be able to engage its audience, providing personalized content depending on the participants' interests. Prior content knowledge acquired through the interaction with other robot guides should be taken into account while selecting a specific topic to talk on. This requires the seamless cooperation of multiple robots exchanging participant's profiles.

This paper focuses on the distributed collaboration of multiple heterogeneous robots (receptionist, companion) welcoming and guiding visitors through a building. The described approach presents two main contributions: the formal definition of a person and robot context, and the location transparency of robot behaviours.

Firstly, several tour topics, each with content on different locations, are defined using an ontology. These semantic descriptions enable a formal definition of human and robot profiles including topics of interest and prior content knowledge, depending on which the robot guides select a specific topic to talk on. The novelty is in the focus on the optimal exchange of information between the robots. Ontologies define a common language for multi-robot interaction, enabling a minimal selection of new and relevant information to be exchanged between the robots. The tour content is optimized through the autonomous transfer of participants by the robots whenever they are in each other's proximity. In this way the robots can, but are not required to, possess the same knowledge and are able to learn from each other. Evaluation of the implemented content delivery and optimization algorithms presents a 90% content coverage of interesting topics for the individual participants.

Furthermore, the actual robot communication is realized through an approach that spreads the individual behaviours of each robot to the whole team. This aggregation is done transparently so the robots share the same set of behaviours, which are either performed by each robot itself or require cooperation with other robots. The result is the execution of a complex task regardless of which robots are used in the process. The described modular solution addresses issues of communication between robots with different operating systems, programming languages and integration with the host infrastructure on which it operates. The concept is validated on an application allowing a receptionist robot to communicate with multiple companion robots providing a guided tour of a building to several visitors.

The remainder of the paper is structured as follows: Section 2 reviews related work on multi-robot collaboration and existing robot guides. Section 3 describes the general concept of collaboration between multiple heterogeneous robots. We define two main challenges: location transparency of robot behaviours, described in Section 4, and formal context definition used during the actual communication between robots and participants, detailed in Section 5. The proposed communication protocol and tour planning algorithms are validated in Section 6. Finally, concluding remarks and suggestions for future work are presented in Section 7.

2. Related Work

The concept of a robot tour guide is not a new one. The Minerva tour guide [3] was successfully exhibited in the Smithsonian Institution focusing on safe navigation in unmodified and dynamic environments, and short-term human-robot interaction. In [4] a companion robot (CoBot) escorts a visitor around a building and performs tasks such as providing schedule notifications, directions to locations, or information on points of interest, and fetching water and coffee. Humanoid robots such as TOURBOT [5] and Robotinho [6] adopt the use of facial expressions. Robotinho interacts with people using multiple modalities such as speech, emotional expressions, eye-gaze and human-like arm and head gestures. Similar to CoBot, it supports omnidirectional walking, self-localization, mapping, obstacle avoidance and path planning. In addition to on-site museum and exhibition tours, TOURBOT offers guided tours to Web visitors. Operating as the user's avatar, the robot accepts commands over the Web that directs it to visit specific exhibits communicating the imaged scenes.

The Santander Interactive Guest Assistants by YDreams [7] guide visitors to their destination. The bots use RFID tags, gyroscopes and odometers to determine their position and 16 sonar sensors to locate objects (such as the human they are guiding) while moving. RFIDs and a wireless sensor network are adopted by [8], supporting tours by multiple robot guides. The independent tour groups consist of a robot leader and several participants. A group-guiding protocol uses sensor nodes to track leaders' locations and maintain paths from members to leaders. A member may ask where his/her leader is and a leader may 'recall' his/her members.

A greater challenge is supporting robot interactions enabling the execution of common tasks. Existing centralized approaches select the best robot to execute a specific task. The network robot platform in [9] determines the most suitable robot for executing a specific service by comparing information on users, robots and services. An area management gateway controls the service execution by coordinating the robots in performing interdependent tasks. The approach in [10] automatically generates a functional configuration of a network robot system performing a given task. This requires actual deployment of the configured application on the robotic network activating the necessary functionalities and setting up the channels between them.

An important aspect of the coordination of multiple robots is area partitioning. In [11] and [12], while the robots are performing continuous area sweeping in respective cleaning tasks, adaptive negotiation methods dynamically partition the area.

The proposed tour planning algorithm in this article partitions a building by optimizing the content delivery by multiple robots to visitors while minimizing the crossing of paths. The robots personalize the guided tour for their participants. The constructed guided tour ontologies define user profiles including topic interests and prior knowledge, which is updated during the tour. The robots optimize the delivery of content in order to engage as many of the participants as possible and at the same time show the building. The amount of provided content is increased through the automatic transfer of participants between the robots whenever they are in each other's proximity and decide that a different guide can provide for more interesting content to certain participants. We focus on a minimal selection of the relevant data to exchange, which is enough to reproduce the same participant information in the other robots.

The key idea of reusing knowledge is fundamental to our approach and a prevailing concept in Cloud Robotics [13]. It provides virtually unlimited resources, alleviating the limitations of robots. For example, the Google Goggle[1] application allows the user to send a photograph of an object and, if it has been previously processed by someone else, the object is recognized. The cloud also stores knowledge and models. The RoboEarth project [14] describes how an articulated arm equipped with sensing capabilities creates a model to open a drawer. Afterwards, another articulated arm with rudimentary sensors can request the information previously stored in the cloud and use it to open the drawer by adjusting the model to its actuator skills.

1 http://www.google.com/mobile/goggles/

The robot communication module we present in this article also aims to reuse the behaviours of other robots, reaching a higher level of cooperation based solely on reusing knowledge from the cloud. The term Robot as a Service [15] was created using the concept of Service-Oriented Architecture, which provides a communication mechanism through standard interfaces and protocols. The idea that each robot maintains a common layer for offering services is shared with the work presented here. However, our approach makes transparent the fact that the services may require communication with other robots, achieving an even higher level of abstraction.

Figure 1. The receptionist exchanges information with the multi-robot guide and provides instructions to the visitors to meet their guide.

3. General Concept

The main objective of this research is the design and implementation of strategies supporting a multi-robot receptionist and guided tour system for visitors. Fig. 1 describes the general concept of the distributed robot receptionist and tour guide. When a group of people arrive at the receptionist's desk, they provide personal information to the robot receptionist Tank. Tank registers this and calls on an available robot tour guide, CoBot. While Tank instructs the visitors on where to meet up with CoBot, their guide arrives at the prearranged location.

Due to the distributed interactions between the active robot guides, information on several topics (architecture, history, research, etc.) of a building is provided to several visitor groups. The discussed topics are selected depending on the participants' prior knowledge and interests, covering as much of the building as possible to reduce the chances of meeting other groups. During the tour, robots exchange context information on their participants and the provided tours, and if necessary swap participants if they can benefit from more interesting topics.

In order to conduct the experiment, two different robot types are used. The first is Tank, discussed in [16-18]. Tank is a robotic receptionist located in the Newell-Simon Hall building at Carnegie Mellon University. Tank's head is a graphically rendered 3D model displayed on a flat-screen LCD monitor mounted on a pan-tilt unit. It uses speech capabilities to provide useful information to visitors. Tank disposes of facial and emotional

expressions to improve the engagement and quality of interaction, while the user input is captured by a keyboard. Its task here is to meet visitors and offer them a guided tour.

The second member of the team is the CoBot [19]. CoBot is a visitor companion robot able to navigate through a multi-floor environment, transport objects, deliver messages and escort people [20]. It carries a variety of sensing and computing devices, including a camera, a Kinect depth camera, a Hokuyo LIDAR, a touch-screen tablet, microphones, speakers, and wireless communication equipment. Its assigned task in this work is to provide a tour of a building to a group of visitors while exchanging information with other CoBot guides.

The following two main requirements are taken into account in detail during the design of the distributed multi-robot receptionist and tour guide:

1. *Behaviour location transparency* allowing for the execution of a robot behaviour without knowing where it is located (Section 4).
2. *Formal context definition* used as communication language for exchanging visitor information between robots (Section 5).

4. Multi-Robot Task Module

The cooperation between multiple heterogeneous robots achieving a complex task such as receiving and guiding visitors requires the seamless integration of the various robot behaviours. For this purpose the Multi-Robot Task Module (MRTM) is developed to support the communication issues between the different robots. MRTM follows a distributed approach and, accordingly, each robot should run an instance of it. The novelty of this work is in presenting MRTM as a tool to encapsulate in each robot all the behaviours offered by a robot team together. The behaviours do not necessarily have to be implemented within the module itself, in contrast to the interface to access those behaviours. The main task of the MRTM module is to invoke the behaviour that has been requested, either by executing it on the robot itself, or by transparently requesting the behaviour of another robot's MRTM component. Fig. 2 displays the internal structure of the MRTM, whose main components are explained below.

4.1 MRTM components

All communication between the robots is handled by the *Internet Communications Engine* (ICE) [21], which is an object-oriented middleware used in distributed systems. Its main virtues include multi-platform and multi-language support and efficiency. An *ICE object* is an entity in a local or remote robot that executes client requests. Its interface declares a set of operations (behaviours) that can be invoked by clients. The operations are declared using the Slice language, which is independent of a specific programming language.

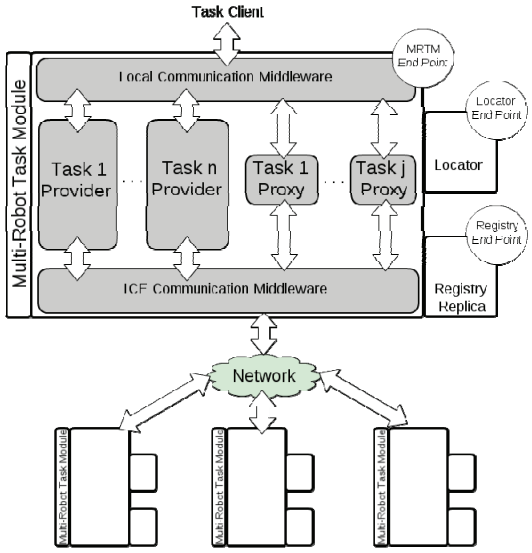

Figure 2. Overview of the Multi-Robot Task Module.

In MRTM, an *ICE object* is realized by creating a class, a *Task Provider*, which implements its interface. This *Task Provider* consists of related behaviours for a given task. Their mission is to trigger the execution of a particular behaviour on the robot. Inside MRTM, there are as many *Task Providers* as local behaviours that the robot supports.

Similar to the *Task Provider*, which initializes the execution of local behaviours, the *Task Proxy* runs behaviours remotely requiring execution by other robots. In order to implement each *Task Proxy* an indirect proxy to an MRTM that is capable of performing the specific behaviour must be acquired, and the remote invocation must be performed using the indirect proxy as an object and the behaviour as its method. Consequently, a *Task Proxy* emulates a specific *ICE object*'s interface and its available operations. The proxy is just a binding in the form of an object, which hides communication with another MRTM. In this work we use indirect proxies, which do not contain any addressing information. In order to find the correct server for a specific robot behaviour, the client-side ICE runtime passes the proxy information to a *Locator* service, which interrogates a *distributed Registry*. This *Registry* associates behaviours with specific robots.

The *ICE Communication Middleware* (ICM) combines all operations related to ICE. In addition, this sub-module is responsible for resolving indirect proxies to perform remote invocations and receive calls from other robots.
In order to receive requests on a given MRTM, they must be called by a different MRTM. When a particular behaviour is requested, if it cannot be executed by any local *Task Provider*, the corresponding *Task Proxy* is used.

When a client or application invokes a specific behaviour, MRTM receives the request through the *Local Communication Middleware (LCM)*. The implementation of this component depends on the internal communication system of the robot. For example, using ROS [22], this component is a node on the robot and its interface forms a set of ROS services. In short, LCM's function is to integrate into the host and serve as an infrastructure gateway for clients of robot behaviours.

4.2. Behaviour location transparency and team coordination

MRTM supports *behaviour location transparency* by acting as a wrapper hiding the location of the requested behaviour (the robot responsible for it) and using a local call. From a developer's perspective, who wants to create a behaviour using other robot behaviours, the degree of code complexity obtained is lower thanks to MRTM. Team coordination is implicitly performed by executing local calls. The ICE framework within MRTM triggers the remote execution of the required behaviour in the suitable robot. The concept is close to RPC, but specialized for robot behaviours.

MRTM links each *Task Proxy* to a specific robot, not currently allocated to a task. The novelty of *behaviour location transparency* is that it presents an abstraction invisible to the human user. When a user requests a specific functionality that is not supported by the robot, it will respond by using a *Task Proxy*. This *Proxy* invokes a remote robot whose reply is indistinguishable from a local reply from the user's perspective. Section 6.1 details a concrete scenario where MRTM handles the interactions between several heterogeneous robots.

5. Multi-Robot Interaction Language Definition

5.1 Guided Tour Modules

Additionally to the communication specification between the various robots, a formal definition of a common language understood by the robot guides is required in order to provide for a personalized tour of a building. Algorithms responsible for the optimal exchange of context information between robots assist with the selection of a specific topic to talk on based on the participants' interests and knowledge.

The main building blocks of the proposed approach are presented in the layered design in Fig. 3. At the lowest level is the *Inference Engine*, capturing the robot and user profiles in an ontology and inferring new knowledge from their interactions. These semantic definitions are encapsulated by corresponding objects in the *Semantic Concepts* layer (locations, topics, robots and tour participants). This additional layer enables switching between different ontologies modelling users, robots and tours, which requires only these specific objects to be updated without affecting the rest of the implementation. The actual planning algorithm for the guided tour is implemented by the *Tour Planner*, which invokes the *Robot Collaboration* component responsible for the communication between the different robots. The whole is enhanced with a *Robot Guide Interface* that visualizes the robot and participant profiles, as well as tours provided, and enables the manual adaptation of the robot profile and the addition/deletion of participants.

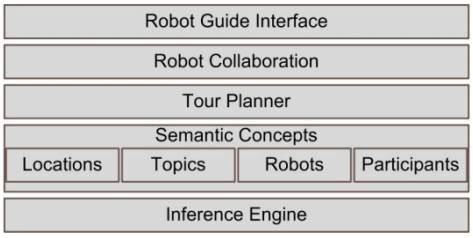

Figure 3. Main architectural components enabling the collaboration between multiple robot tour guides.

5.2 Ontological Guided Tour Definition

As the guided tour is rather information intensive, it requires capturing the gathered knowledge and interactions between the different parties in a machine-processable common vocabulary, also known as an ontology. A typical ontology language is OWL (Web Ontology Language) [23], which is a well-defined vocabulary for describing a domain with a foundation in description logics. A guided tour ontology is created using the Protégé Editor [24]. This editor provides support for OWL, RDF and XML schemas, making it possible to easily design ontologies through a graphical interface. Additional knowledge is captured using SWRL (Semantic Web Rule Language) expressions and built-ins (SWRLB), such as comparisons (equal, less/greater than), mathematical functions (add, subtract, multiply, divide) [25], etc. SWRL expressions are used for coding procedural relations in the form of rules.

Our approach brings the required guided tour concepts under an OWL robotics ontology. Fig. 4 illustrates the definition of a *Content* concept at a *Location* on a specific *Subject* (topic) with a certain *LevelOfDetail* and possibly additional details. The *Content Subject*, covering topics such as architecture, history and research, defines the robot's and person's interests. Based on the *LevelOfDetail*, *introduction* or *details*, the robot provides new information whenever it passes through the same *Location* using the *hasDetails* property of the *Content*. The actual information as provided by the robot is defined by the *tourInformation* property.

The following example defines *introductory content* on the 'Robotics lab' with additional details on 'Robotics people' and 'Robotics projects'. It is classified as a *research subject* at *office* 'Office-7412'.

'Robotics lab'
hasDetails 'Robotics people', 'Robotics projects'
hasMetadata 'introduction'
hasMetadata 'research'
onLocation 'Office-7412'
tourInformation "Research on the scientific and engineering challenges of creating teams of intelligent agents in complex, dynamic, and uncertain environments."

When a robot talks on a specific content, a *Guided Tour* concept is created (Fig. 5), consisting of this *Content*, the robot *tour guide* and its *participants* and a *timestamp*.

The example below defines a 'Robotics tour' on the 'Robotics lab' covered by 'Cobot 1' with two participants.

'Robotics tour'
hasContent 'Robotics lab'
hasTourGuide 'Cobot 1'
hasTourParticipant 'Anna','Carlos'
timestamp "2011-12-18T07:30:00"

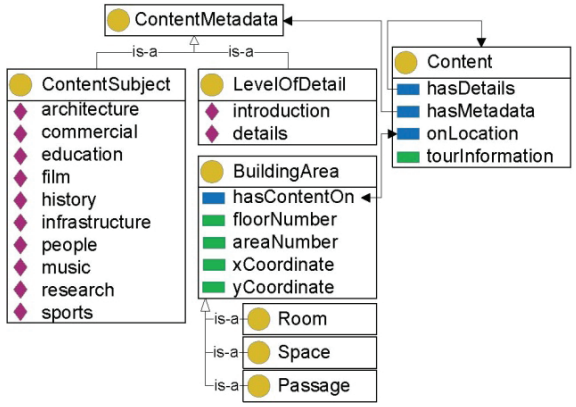

Figure 4. Semantic representation of content on a specific subject (topic) at a location with a certain level of detail.

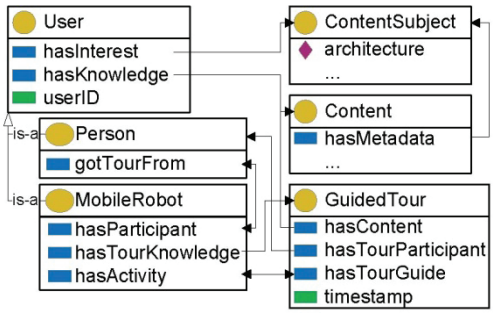

Figure 5. Relationships between tour guides, participants and a specific tour.

Instead of explicitly specifying the tour guide and participants we define the tour's *hasTourGuide* property as an inverse property of the robot's *hasActivity*. The **Inference Engine** will automatically infer that if *hasActivity (CoBot 1, Tour A)* then *hasTourGuide(Tour A, CoBot 1)*. The tour's *hasTourParticipant* is inferred through the following SWRL rule stating that a tour participant is a participant of a robot giving a tour on a specific topic.

MobileRobot(?robot), GuidedTour(?tour), User(?user),
hasActivity(?robot, ?tour),
hasParticipant(?robot, ?user)
-> hasTourParticipant(?tour, ?user)

As the robots provide content depending on their participants' knowledge, it should be possible to request previously acquired data. In order to infer this information, a rule stating that a tour participant has knowledge on the content of the tour is defined.

GuidedTour(?tour), User(?user), Content(?content),
hasTourParticipant(?tour, ?user),
hasContent(?tour, ?content)
-> hasKnowledge(?user, ?content)

Additionally, a property *hasTourKnowledge* is specified between a robot and a tour concept. It is used whenever robots need to exchange information on their participants' knowledge. For example 'Cobot 1' has provided the previously defined 'Robotics tour' to its participants 'Anna' and 'Carlos'. At some point, 'CoBot 2' will want to update its data in order to take over participants from 'Cobot 1'. At this point 'Cobot 1' will simply request data on which tours 'CoBot 2' has knowledge of. It will see that the 'Robotics tour' is new, as it does not have the defined property *hasTourKnowledge(CoBot 2, Robotics tour)*. Consequently, 'Cobot 1' will provide the 'Robotics tour' information to 'CoBot 2' and add the *hasTourKnowledge(CoBot 2, Robotics tour)* property to its knowledgebase. As the 'Robotics tour' defines the *tour content*, *participants*, *robot guide* and *timestamp*, it can reproduce the knowledge of 'Cobot 1' and its participants. Using the defined SWRL rule *hasKnowledge* with information on *hasContent* and *hasTourParticipant* from the definition of the 'Robotics tour', it will infer that 'Carlos' and 'Anna' know more on the 'Robotics lab' as visualized in the following way:

GuidedTour(Robotics tour), User(Carlos), Content(Robotics lab),
hasTourParticipant(Robotics tour, Carlos),
hasContent(Robotics tour, Robotics lab)
-> hasKnowledge(Carlos, Robotics lab)

5.3 Exchange of context between robots

In order not to impose any restrictions on the robots' configuration, the tour guides have different knowledge-simulating guides with diverse specialization and knowledge on their participants. The semantic definition of a participant includes his profile (name, age, picture, place of origin), which is collected by the robots. During a tour the participants also exchange this personal content with their guides, which in turn is transferred between the robots. This enables a dialogue between a robot and a participant. If 'Robot A' asked for a country of origin, 'Robot B' can ask in which city the person was born when the participant switches guides. Therefore, the intelligent

exchange of context information between the robots is of the utmost importance as the guided tour is executed in a distributed manner. This is supported by the **Robot Collaboration**.

Instead of sending all the participants' knowledge and interests each time, which is already partially known by the other robots, a selection is made of the relevant new information. As has been mentioned, a robot only sends data on the new tours to the other robots (*hasTourKnowledge* property). Due to the use of the same ontology and rules in all the robots, the data residing in one robot is reproduced, reducing the amount of information that needs to be sent.

The robots' knowledge on their participants is vital for delivering an interesting selection of content. In the best case, if a robot is unable to deliver any more interesting information to a participant but another robot *is* able to do this, there should be an automatic transfer of this participant to the new robot. Fig. 6 presents two possible solutions (cases A and B), depending on which robot triggers the transaction. Let us assume that a participant is transferred from 'Cobot 1' to 'CoBot 2'. The next sections describe the two developed solutions: (A) giving participants, and (B) taking over of participants.

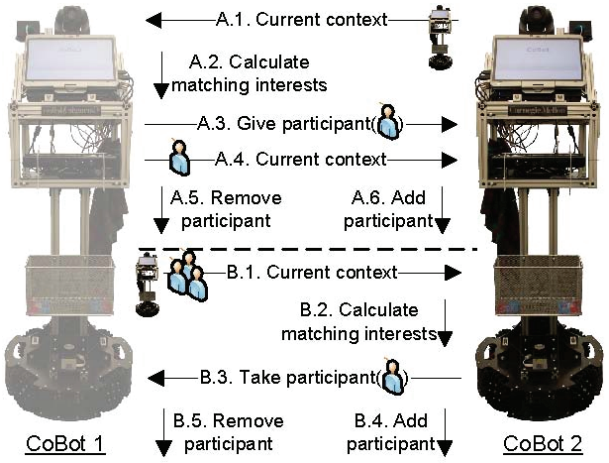

Figure 6. Robot interactions during participants' exchange.

5.3.1 Giving participants (A)

If 'Cobot 1' decides that it cannot provide enough new and interesting *content* to a person, it automatically gives the participant to another robot with a better *interests* match. In order to do so, the robot needs information on the other robots' interests and knowledge. The robots exchange the following information:

1. 'CoBot 2' sends its current context, consisting of its ID, topic interests and the content knowledge it has already provided to its participants.
2. 'Cobot 1' calculates which robot has a better interests fit with its participants.
3. 'Cobot 1' sends a *giveParticipant(userID)* message to 'CoBot 2' for each better-matching participant.
4. 'Cobot 1' queries its data on the tour knowledge (*hasTourKnowledge* property) of 'CoBot 2', sending only the new tours to 'CoBot 2'. If the exchanged participant is not known by 'CoBot 2', 'Cobot 1' adds its profile to the exchanged context.
5. 'Cobot 1' deletes the participant from its list.
6. 'CoBot 2' adds the participant to its list.

The data exchanged includes the robot's profile, new tour data and, optionally, the new participant's profile.

5.3.2 Taking over participants (B)

If 'CoBot 2' wants to check if the other robots have participants that better match its *interests* and *knowledge*, it needs information on the robots and their participants. The following transactions are performed:

1. 'Cobot 1' queries its data on the tour knowledge (*hasTourKnowledge* property) of 'CoBot 2', sending only the new tours to 'CoBot 2' as its current context. Additionally, any missing profile information on 'Cobot 1' and its participants is also exchanged.
2. 'CoBot 2' calculates which participants better fit its profile instead of the owner's.
3. 'CoBot 2' sends a *takeParticipant(userID)* message to 'Cobot 1' for each better-matching participant.
4. 'CoBot 2' adds the participant to its list.
5. 'Cobot 1' deletes the participant from its list.

The information exchanged includes the new tour data and the robot and participant profiles. One advantage of this exchange is that more participants are taken over without extra context function calls being needed.

5.3.3 Calculate matching interests

Calculation of the matching interests between a participant and a robot during participant exchange is implemented in various ways with different complexity and correctness:

- *Interests:* comparison of the two robots' interests and the participant's interests. The robot that has more matching interests with this participant is selected.
- *Own Knowledge:* remove from the matching interests topics and content known by the three parties. This approach considers content already covered by the robots during their tours and prior participant knowledge. It is possible that, although the new robot has more matching interests, it will never talk about them after the participant exchange as they were already discussed with other participants.
- *Other Participants Knowledge:* remove content known by the rest of the robots' participants. This knowledge

is taken into account as the provision of future content depends on all the participants.

A trade-off should be made between the complexity and the correctness of the calculation of the matching interests between the robots and the participants, as with each solution there is additional information to be considered.

5.4 Topic and area division between the robots

Each time a topic is finished the robot has to select a new one based on its participants, other robots and the building area in order to guarantee an optimal topic division and area coverage by the robots. Only one topic is covered at a time, focusing on showing the building instead of standing too long at the same location. The robots dynamically change navigation paths in order to stay out of each other's proximity. Especially for big groups, even in real-life scenarios with human guides, it is always tedious when guides cross over, as this results in too little space and too much noise to optimally enjoy the tour. The topic selection algorithm implemented by the **Tour Planner** is presented in Fig. 7.

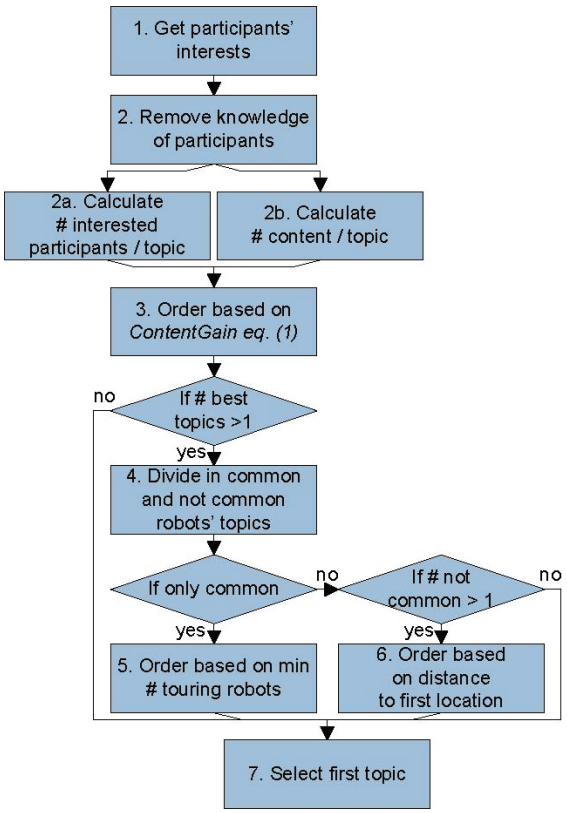

Figure 7. Topic selection based on the number of interested participants, new locations and content, common topics with other robots and the distance to the first location of a topic.

1. Starting from the topics known by the robot, only the topics interesting for the participants are retained.
2. Acquired content knowledge of the participants is removed using stored profiles from prior robot tours. The result is the removal of entire topics or specific topic content. Next, two lists are created based on:
   a. the number of interested participants per topic;
   b. the maximum amount of content per topic.
3. Taking into account the number of interested participants (P) and the amount of available content for each topic (I+D), defined by the tour ontology, order the topics based on the *ContentGain (CG)* Eq. (1) where I is the amount of *introductory* content and D the amount of *additional* content (*LevelOfDetail* concept in Fig. 4). During the actual tour, the robot provides detailed information on a specific location if more than 50% (flexible parameter) of the participants are interested, resulting in the two parts of Eq. (1). The resulting list reflects the amount of new knowledge for all participants acquired if the robot selects this topic. At this point the best topic (maximum *ContentGain*) is selected (Step 7).

$$CG = \begin{cases} P \times (I + D), \text{ if } P \geq 50\% \text{RobotGroup} \\ P \times I, \text{ otherwise} \end{cases} \quad (1)$$

4. If there are several topics with the same *ContentGain*, the robot looks at the topics covered at this moment by the other robots. The list with equivalent topics is split into common topics and not-common topics.
5. If there are only common topics, the list is ordered based on the minimum number of robots and the first one is selected. It is still possible that there are several equivalent topics, which can be ordered based on the existence of empty floors or distance from other robots; as it is a dynamic environment this is not considered here.
6. If, on the other hand, there are topics not overlapping with other robots, the closest topic is selected.
7. Select the best topic (maximum *ContentGain*).

Once a topic is selected the robot uses a simplified algorithm to organize the content delivery in order to optimize performance. If a topic is selected common to other robots, the starting point is not necessarily the closest location, but the farthest point (elevator, floor) from the other guides. This prevents robots from following the same tour path. During the tour, the robots exchange information on their location and adapt the next tour point in case of possible overlap. Robots with low ID number (string comparison) keep their current path while those with higher ID recalculate their next point. As mentioned, if at a specific location more than half of the participants are interested in the topic, the robot not only provides a basic introduction but also additional details if available.

The basic example below clarifies the described algorithm in Fig. 7 for 'Cobot 1' with five participants.

1. Retained interesting topics (amount of available content): *research* (9), *architecture* (7), *people* (11), *history* (10).

2. Removal of prior participant knowledge results in: *research* (7), *architecture* (6), *people* (6).

a. Order based on the number of interested participants: *architecture* (P=3), *people* (P=3), *research* (P=2).

b. Order based on the amount of content per topic: *research* (7 with I=4, D=3), *architecture* (6 with I=4, D=2), *people* (6 with I=3, D=3).

3. Order based on *ContentGain*: *architecture* (18), *people* (18), *research* (8). *Research* is removed from the list.

4. Common topics to other robots: *architecture* (covered by 'CoBot 2'); not common: *people*.

7. Select topic on *people*.

6. Experimental Results and Analysis

6.1 Distributed Robot Behaviour Transparency

In order to illustrate and validate the proposed MRTM design developed in Section 4, we applied it to a team of robots performing a distributed visitor reception and guidance task. The task requires a receptionist to welcome visitors and ask for their personal details. The receptionist retrieves and offers the location of the available CoBot able to perform a guided tour, or suggests the option of being escorted to a given location.

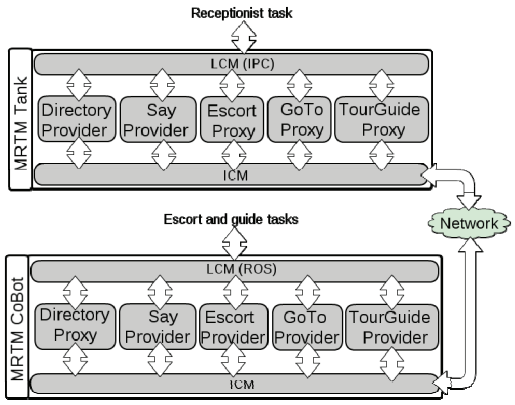

Figure 8. MRTM architecture for the Tank and CoBot cooperation tasks.

Fig. 8 shows the overall architecture of the experiments (the elements **Registry** and **Locator** have been deliberately omitted to simplify the diagram). All agents have the same interface, which allows the deployment of *Directory*, *Say*, *Escort*, *GoTo* and *Guide* behaviours. While in this scenario the application of user interaction was only deployed on Tank, any other robot could technically have offered the same functionality.

The LCM module of the CoBot is integrated into a specific ROS node responsible for managing its behaviour. In turn, the LCM module of the receptionist Tank is included in a IPC[2] (Inter Process Communication) node.

The Slice interfaces shared among the robots of this experiment are shown below. The *Directory* requests the office of a given person. *Speech* converts a text message into a voice message. *Escort* and *Navigation* include the methods *escort* and *goto*, which select a specific office and a time window and return a task ID. This value is used to cancel the booking by invoking the *cancelTask* method. While the *goto* behaviour initiates the CoBot's movement towards the destination, the *escort* behaviour moves Cobot towards an elevator's area (the one located on the same floor of the visitor's destination). Once the visitor arrives at the meeting point and informs CoBot on his/her presence, CoBot escorts the visitor to the desired location.

```
module MultiRobotServices {
  interface Directory  {
    string people2office( string personName ); };
  interface Speech {
    void say( string message ); };
  interface Escort {
  string escort( string room, string startDate,   string startH, string startM,
string startP, string   endDate, string endH, string endM, string endP );
              int cancelTask( string taskId ); };
  interface Navigation {
    string goto( string place, string startDate,   string startH, string startM,
string startP, string   endDate, string endH, string endM, string endP );
              int cancelTask( string taskId ); };
  interface TourGuide {
    string startTour ( string startPlace, string startDate,    string startH,
string startM, string startP, string    endDate, string endH, string endM,
string endP ); };
};
```

In the first test case, a visitor is escorted to an office using MRTM. After greeting and interacting with the visitor, Tank uses its **Directory Proxy** to consult the directory and ask for the destination. Note that Tank does not even know which robot or agent actually implements this service. The receptionist application invokes the method *person2office*. Once the visitor is informed of the office number, Tank offers the visitor the option of being escorted. If the proposal is accepted, Tank uses the *escort* method offered by its MRTM to start escorting the visitor.

As described above, the MRTM module takes over and triggers the *escort* behaviour using the **Escort Proxy** of Tank. CoBot has a scheduler for requesting different behaviours and when they should be executed by the robot. The **Escort Provider** of the CoBot robot makes a reservation for the *escort* task before offering visitors the option of being escorted. If the reservation is successful, the receptionist Tank offers the possibility of escorting the visitor. Then, if the visitor rejects the offer, Tank uses the *cancelTask* method to remove that task from the scheduler. A demonstration of this experiment is available on this site[3] with different video resolutions.

The second test case describes the sequence of steps to start a tour using MRTM. After the same greeting and

2 http://www.cs.cmu.edu/~ipc/

3 http://gsyc.es/~caguero/videos/caguero-cobot/

interacting phase with the visitors discussed above, Tank uses the CoBot interface to book an available tour guide. Tank invokes the *startTour* method offered by its MRTM to call a CoBot guide as if it is implemented in itself.

As described above, the MRTM module takes over and triggers the *startTour* behaviour using the **TourGuide Proxy** of the Tank robot. Cobot has a scheduler for requesting different behaviours and when they should be executed by the robot. The **TourGuide Provider** of the CoBot makes a reservation for the *goto* task before offering visitors the tour. If the reservation is successful, the receptionist Tank offers information on where to find the CoBot and start the tour. All the details of the multi-robot tour guide are presented in Section 6.2.

The resources consumed by MRTM were measured during the experiment. CPU overhead is completely negligible, 40MB of memory was consumed and there was no continuous bandwidth used. The spikes on the bandwidth consumption were at a maximum of 1Kb/sec, and occurred during the CoBot requests.

As described in Section 3, a module providing *behaviour location transparency* is one of the contributions of this work. In the experiments, this aspect is illustrated by Tank, which invokes the *escort* and *tourGuide* behaviours. These behaviours cause the remote execution of tasks on another robot (CoBot). However, Tank runs a set of apparently local behaviour calls. As described in Section 4, MRTM takes care of all the internal communication aspects and hides the execution of remote behaviours as local calls, providing the desired *behaviour location transparency*.

6.2 Personalized selection of interesting content

A major challenge of the distributed tour guides is in providing a personalized selection of interesting content to several groups of participants depending on their prior knowledge. In order to measure the actual content delivered by the robots one should compare the content provided by the robots with the amount of content that is relevant for the participants.

In order to provide for more extensive results, the actual evaluation is carried out in a simulation where the program is executed reproducing the robot movements. An ontology generator is developed that takes into account parameters, such as number of robots and participants per robot, and content distribution generates content for all the possible rooms in a building and a list of the robot and participant interests. During the actual evaluation, two robots are defined, each with five participants. The robot and participant interests are selected randomly out of a predefined list of topics. Using the same list of topics, the offices of a nine-storey building are automatically enriched with content. Each office has a 50% chance of having a specific content with a 50% chance of details, and another 50% chance of having additional content on another topic with possible details. In total, 10 test sets are generated in order to obtain average results.

For each of the 10 test sets, the following five experiments are executed measuring the optimization of the content delivery (*ContentGain*) by the robots to their participants:

Random: randomly (alphabetically) selected topics.
Ordered: topics are selected using the *ContentGain* optimization algorithm (Eq. (1)) in Section 5.
Exchange of participants between the robots using the taking over scenario described in Section 5.3.2:
- *Interests*: once during the start of a robot tour maximizing the number of matching interests with the robots.
- *Own knowledge*: each time the robots are on the same floor based on the amount of new knowledge that can be provided by both robots.
- *Others knowledge*: each time the robots are on the same floor based on the comparison of new knowledge provided by both robots taking into account the knowledge of the original robot group at the moment.

As mentioned in Section 3 a major contribution of the presented framework is the *formal context definition*, which is used as a communication language between the robots. Thanks to the formally defined Guided Tour ontology in Section 5.2, the robots not only have knowledge on each other's tours and participants but are also able, based on this knowledge, to exchange a minimal amount of information that is used to optimize the content delivery and transfer of participants.

The results shown in Fig. 9 and Fig. 10 compare the *ContentGain* of the participants of both robots in respect to the total content knowledge of the robots. The evolution of the tour is measured in by a percentage of the provided content against the total amount of provided content at the end of the tour. Results are averages over the 10 test sets.

Fig. 9 compares the amount of *ContentGain* between a *random* and *ordered* topic selection. Although both provide the same amount of content, the *ordered* solution engages as many participants as possible resulting in a curved line. A one-off exchange of participants based on the number of matching *interests* results in an additional 10% *ContentGain* compared to the solutions without exchange.

The different participants' exchange solutions are compared in Fig. 10. Taking into account the individual's

and the robot group's knowledge results in an additional *ContentGain* of 8 (*own knowledge*)-9 (*others knowledge*) %.
The evaluation of the *ContentGain* shows that robots are able to engage people in an interesting robot tour just like a real tour guide using a formal notion of interests and knowledge. Unlike standard tours, through an exchange of participants we propose a more personalized tour where robots not only interact with their group members but also with each other in order to optimize the distribution of people into common groups of interests. This solution supports robot groups starting at different times with a variety of participants converging into groups with similar interests.

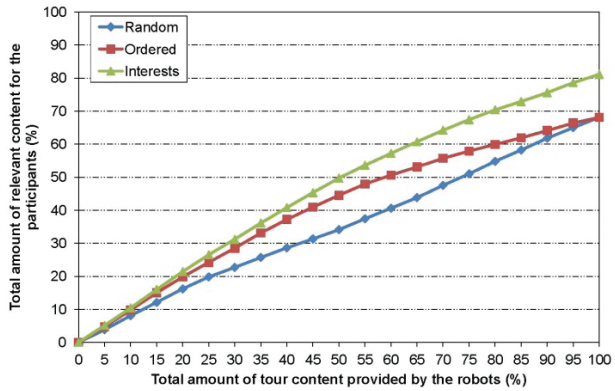

Figure 9. Comparison between *random* and *ordered* based on people's interests, topic selection, and exchange of participants optimizing robot-participant interests match.

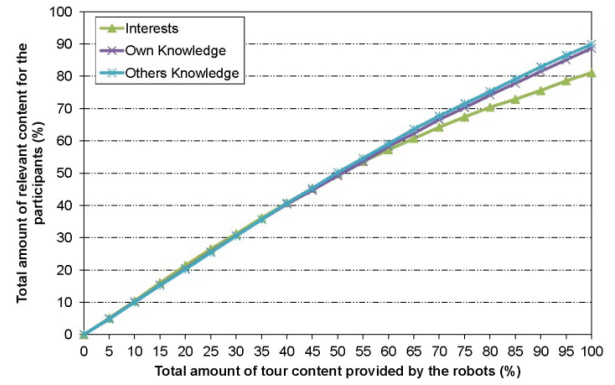

Figure 10. Comparison of amount of content delivery during participant exchange based on matching interests, known content and robot group knowledge.

Additionally, we evaluated the performance of the reasoning on the Guided Tour ontology from Section 5.2 together with the 10 generated building tours for simulation. Depending on the defined number of semantic concepts, properties and instantiated individuals, Table 1 provides metrics on the ontology classification performed by the **Inference Engine**. The main goal of the framework is to provide for a minimal amount of data exchanged between the robots, in order to reproduce each other's knowledge. There are two cases that need to be distinguished. The first is when a robot behaves independently of the other robots. In this case there is no time loss as the application is optimized to keep the required data in the **Semantic classes** in Fig. 3. This encapsulation of the semantic concepts reduces the number of times the ontology needs to be queried. The second case is when robots actually do exchange data, which requires more complex reasoning in order to reproduce all the knowledge on the other robots. In this case it takes less than a second for the reasoner to perform classification on the updated data.

Robotics ontology	
Class count	316
Object property count	212
Data property count	47
Individual count	0
Ontology classification during initialization	848 ms
Avg. ontology classification during execution	320 ms
Generated building tours used for simulation	
Avg. individual count for 10 experiments	1453
Ontology classification during initialization	1035 ms
Avg. ontology classification during execution	836 ms

Table 1. Performance of the reasoning on the ontology.

The main conclusion from human experience is that following a specific robot and being able to switch tour guides makes for a dynamic and interesting experience. One possible disadvantage of the approach is that if robots transfer visitors too much between each other, this will result in too many annoying interruptions of the tour. Interesting future work would be a long-term evaluation of the framework, focusing on exactly this human experience and trying to find a trade-off between content optimization and a minimum number of visitor transfers.

7. Conclusion

This paper presents collaboration techniques between multiple robots welcoming and guiding visitors through a building. Firstly, we focus on the definition of a tour ontology specifying the various topics, available content per topic and the knowledge and interests of the robots and the participants. The robots select topics depending on the participants' interests and prior knowledge, resulting in a personalized tour. Additionally, the robot guides interact with each other, exchanging participant profiles and provided tour information. This cooperation enables the automatic transfer of members each time the robots are in each other's proximity in order to optimize the amount of interesting content. Evaluation of the deployed algorithms for two robot groups presents a 90% content coverage of the topics of interest to the individual participants. In the case where the approach was extended to more robots, preliminary results indicate that the amount of messages exchanged will grow as more robots need to synchronize their knowledge base, whilst at the same time a single robot will receive pieces of information from several robots. This exchange between

more than two robots might slightly slow down the planning process, optimizing which will be an interesting new goal.

A second major contribution is the cooperation among several heterogeneous robots. The presented test cases detail CoBot to CoBot and Tank to CoBot communication. MRTM is proposed as a solution to the problem of remote execution of behaviours with location transparency. Requirements involving localizing behaviours and interoperability between different platforms are tackled using the tools provided by the ICE middleware. This approach allows for the integration of a MRTM module into devices such as robots with different operating systems, with minimal impact on CPU overhead, memory consumption and network bandwidth. Various experiments with robots and humans are conducted successfully for a multi-robot receptionist and several companion robots. The receptionist Tank is ignorant of the location of the directory, which robot is escorting the visitors, or who is able to start a tour. All the low-level details of exchanging information are hidden by MRTM.

Future work will focus on a long-term evaluation of the developed framework, consisting of a large group of robots providing several tours of a building per day to visitors. The validation will provide various metrics on performance and scalability of the semantic reasoning, the number of message exchanges between robots and amount of exchanged data, and visitor satisfaction. Additionally, it might be interesting to support dynamic changes in the ontology. An eventing mechanism will notify robots on new or updated content. Robots can subscribe to these changes and acquire new knowledge at runtime. This enables robots not only to learn new content, but also subscribe to topics of their interest.

8. Acknowledgments

The authors would like to thank the Special Research Fund of Ghent University (BOF) for financial support through A. Hristoskova's PhD grant, and the Computer Science group of Professor M. Veloso for hosting A. Hristoskova and C. Agüero at Carnegie Mellon University. This work is funded by a Fund for Scientific Research fellowship for a stay abroad, Flanders (FWO), by the projects S2009/DPI-1559 and RoboCity2030-II at the Comunidad de Madrid, and by project 10/02567 at the Spanish Ministry of Science and Innovation.

9. References

[1] Schlenoff C., Messina E. (2005). A robot ontology for urban search and rescue, in Proceedings of the 2005 ACM workshop on Research in knowledge representation for autonomous systems, pp. 27–34.

[2] Amigoni F., Neri M. (2005). An application of ontology technologies to robotic agents, International Conference on Intelligent Agent Technology, IEEE/ WIC/ ACM 2005, pp. 751–754.

[3] Thrun S., Bennewitz M., Burgard W., Cremers A., Dellaert F., Fox D., Hahnel D., Rosenberg C., Roy N., Schulte J., et al. (1999). Minerva: A second-generation museum tour-guide robot, in Proceedings of the 1999 IEEE International Conference on Robotics and Automation, vol. 3, pp. 1999–2005.

[4] Rosenthal S., Veloso M. (2010). Mixed-initiative long-term interactions with an all-day-companion robot, 2010 AAAI Fall Symposium Series, pp. 97–102.

[5] Trahanias P., Burgard W., Argyros A., Hahnel D., Baltzakis H., Pfaff P., Stachniss C. (2005). Tourbot and web fair: Web-operated mobile robots for tele-presence in populated exhibitions, Robotics & Automation Magazine, vol. 12, no. 2, pp. 77–89.

[6] Faber F., Bennewitz M., Eppner C., Gorog A., Gonsior C., Joho D., Schreiber M., Behnke S. (2009). The humanoid museum tour guide robotinho, The 18th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2009, pp. 891–896.

[7] YDreams, SANTANDER SIGA, (2010). Available: http://www.ydreamsrobotics.com/projects/.

[8] Chen P. (2007). A group tour guide system with rfids and wireless sensor networks, 6th International Symposium on Information Processing in Sensor Networks, IPSN 2007, pp. 561–562.

[9] Nakamura Y., Machino T., Motegi M., Iwata Y., Miyamoto T., Iwaki S., Muto S., Shimokura K. (2008). Framework and service allocation for network robot platform and execution of interdependent services, Robotics and Autonomous Systems, vol. 56, no. 10, pp. 831–843.

[10] Lundh R., Karlsson L., Saffiotti A. (2008). Autonomous functional configuration of a network robot system , Robotics and Autonomous Systems, vol. 56, no. 10, pp. 819–830.

[11] Ahmadi M., Stone P. (2006). A multi-robot system for continuous area sweeping tasks, International Conference on Robotics and Automation, ICRA 2006, pp. 1724–1729.

[12] Jager M., Nebel B. (2002). Dynamic decentralized area partitioning for cooperating cleaning robots, IEEE International Conference on Robotics and Automation, ICRA 2002, vol. 4, pp. 3577–3582.

[13] Guizzo E., Cloud Robotics: Connected to the Cloud, Robots get Smarter, (2011). Available: http://spectrum.ieee.org/automaton/robotics/robotics-software/cloud-robotics.

[14] Waibel M., Beetz M., Civera J., D'Andrea R., Elfring J., Galvez-Lopez D., Haussermann K., Janssen R., Montiel J.M.M., Perzylo A., et al. (2011). RoboEarth, Robotics Automation Magazine, IEEE, vol. 18, no. 2, pp. 69–82.

[15] Chen Y., Du Z., García-Acosta M. (2010). Robot as a Service in Cloud Computing, in Proceedings of the

2010 Fifth IEEE International Symposium on Service Oriented System Engineering, SOSE '10, IEEE Computer Society, pp. 151–158.
[16] Gockley R., Bruce A., Forlizzi J., Michalowski M., Mundell A., Rosenthal S., Sellner B., Simmons R., Snipes K., Schultz A.C., Wang J. (2005). Designing robots for long-term social interaction, IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2005, pp. 1338–1343.
[17] Gockley R., Forlizzi J., Simmons R. (2006). Interactions with a moody robot, in Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction, pp. 186–193.
[18] Kyung Lee M., Makatchev M. (2009). How do people talk with a robot? An analysis of human-robot dialogues in the real world, in Proceedings of the 27th international conference extended abstracts on Human factors in computing systems, CHI'2009, pp. 3769–3774.
[19] Licitra M., CoBot Robots, (2011). Available: http://www.cs.cmu.edu/ coral/projects/cobot/.
[20] Biswas J., Veloso M. (2011). Depth camera based localization and navigation for indoor mobile robots, RGB-D Workshop at RSS 2011.
[21] Henning M. (2004). A new approach to Object-Oriented Middleware, IEEE Internet Computing, vol. 8, pp. 66-75.
[22] Willow Garage, ROS (Robot Operating system) , (2012). Available: http://ros.org/
[23] McGuinness D., Van Harmelen F., et al., OWL web ontology language overview, (2004). Available: http://www.w3.org/TR/owlfeatures/.
[24] Stanford University, Protégé, (2011). Available: http://protege.stanford.edu/.
[25] Horrocks I., Patel-Schneider P.F., Boley H., Tabet S., Grosof B., Dean M. (2004). SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Available: http://www.w3.org/Submission/SWRL/